BSM 304
İşletim Sistemleri

8. Hafta – CPU Scheduling (CPU Planlama)

Dr. Öğr. Üyesi Onur ÇAKIRGÖZ
onurcakirgoz@bartin.edu.tr

BÖLÜM – 7

Bu bölümde,

- Temel Kavramlar
- Planlama Kriterleri
- Planlama Algoritmaları
- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-First (SJF) Scheduling
- Priority Scheduling
- Round-Robin Scheduling

konularına değinilecektir.

Temel Kavramlar

- CPU scheduling (planlama), multiprogramming çalışan işletim sistemlerinin temelini oluşturur.
- CPU, **prosesler arasında geçiş yaparak** bilgisayarı daha verimli kullanılır hale getirir.
- *Tek işlemcili (tek çekirdekli) sistemlerde*, **bir t anında sadece bir proses çalıştırılabilir**. Birden fazla proses olduğunda, bunlar CPU'nun işinin bitmesi için bekleyeceklerdir.

Temel Kavramlar (devam…)

- Hafızada çok sayıda proses bulundurulur.
- Bir proses herhangi bir şekilde **beklemeye geçtiğinde**, CPU başka bir prosese geçiş yapar.

Temel Kavramlar (devam…)

- CPU **planlama algoritmasına göre** sırası gelen proses,
  - Ready - Hazır Kuyruğundan alınarak,
  - Dağıtıcı (Dispatcher) ismi verilen bir işlem tarafından CPU'ya gönderilir.
- CPU'da, yine proses planlama algoritmasının izin verdiği kadar (*ya **bitene** ya da **belirli bir zaman** geçene kadar*) çalışan program
  - ya biter ve hafızadan kaldırılır
  - ya da **tekrar bekleme sırasına** bir sonraki çalışma için yerleştirilir.

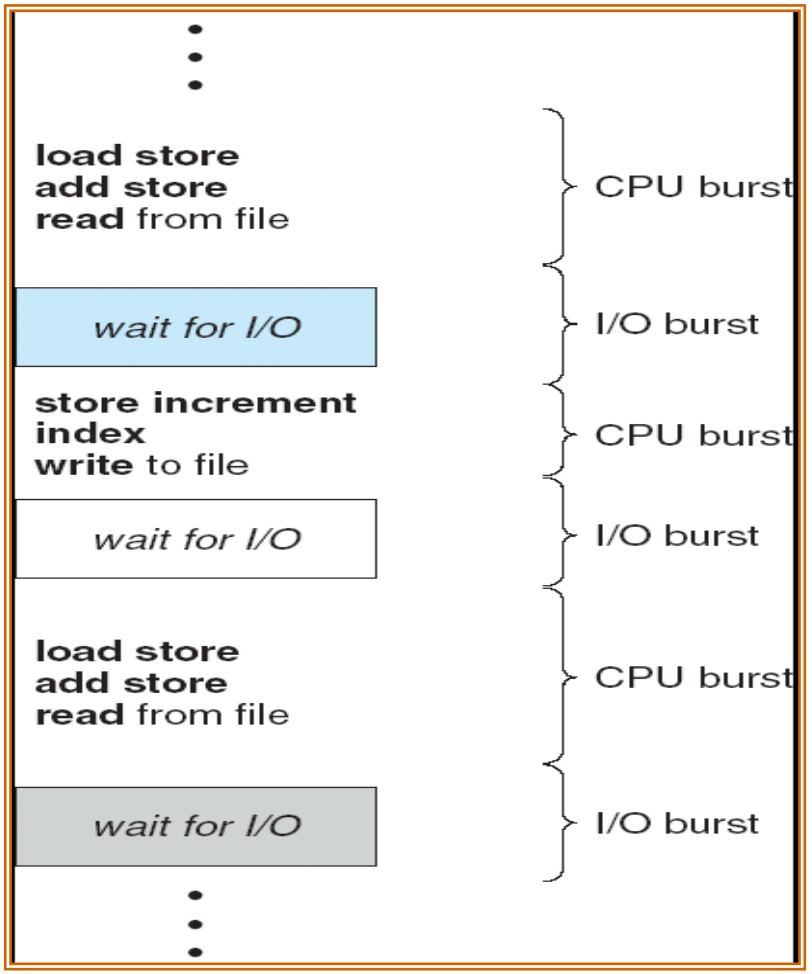
CPU - I/O Burst Cycle

- **CPU-Burst**, CPU'nun bir prosesi işlemek için **I/O Wait gelene kadar** *ihtiyacı olan zaman aralığıdır*.
- Başka deyişle CPU'nun tek proses için *tek bir seferde* harcadığı vakittir. Burst time = execution time.
- Proses işletimi:
  - CPU burst ile başlar ve **sonra** I/O burst gelir bunu başka bir CPU burst ve arkasından başka bir I/O burst bunu takip eder.
  - **Son CPU burst ile program sona erer**.

CPU - I/O Burst Cycle (devam…)

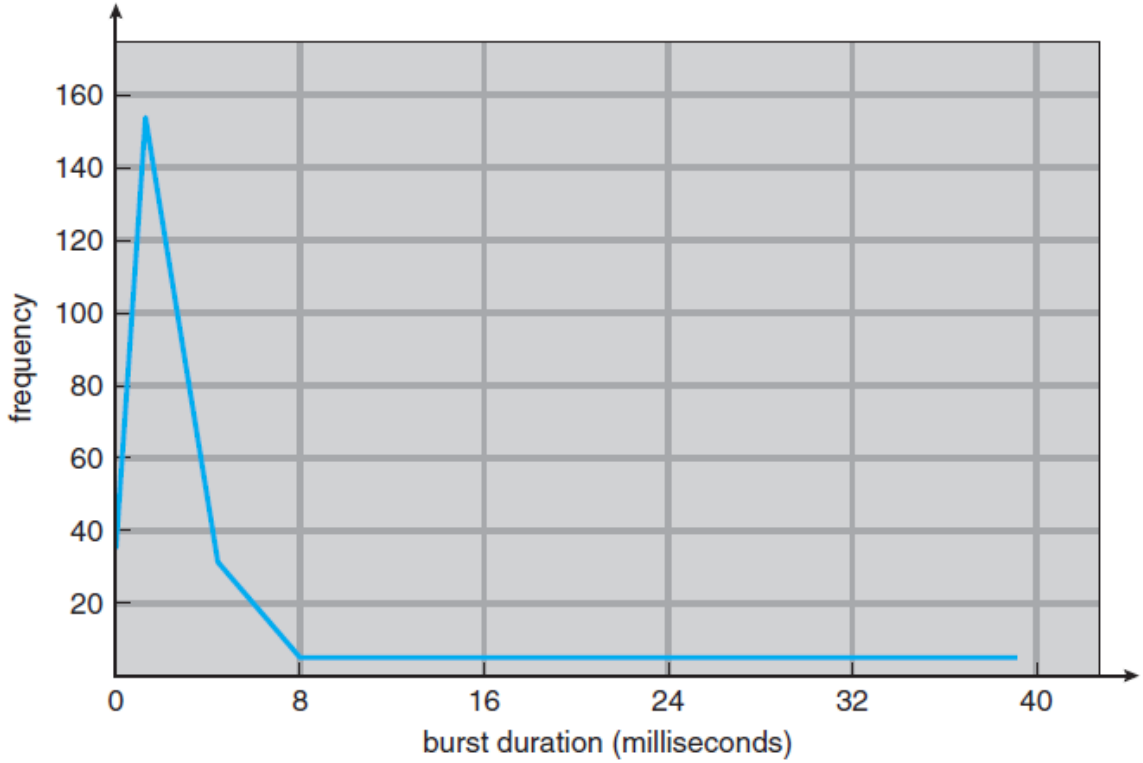
- I/O ve CPU burst arasındaki kullanım **sıklığı ve bekleme**, prosesin
  - **CPU bound** veya
  - **I/O bound**

olması ile önemli oranda ilgilidir.



CPU - I/O Burst Cycle (devam…)

- CPU burst süresi, **prosesten prosese** ve **bilgisayardan bilgisayara** *çok farklı olabilmektedir*. CPU burst süresi **kısa olanlar** **çok sık**, CPU burst süresi **uzun olanlar** ise **çok seyrek** çalışmaktadır.
- I/O-bound proses, birçok kısa CPU burst'e sahiptir.
- CPU-bound proses, az sayıda CPU burst'e sahiptir.

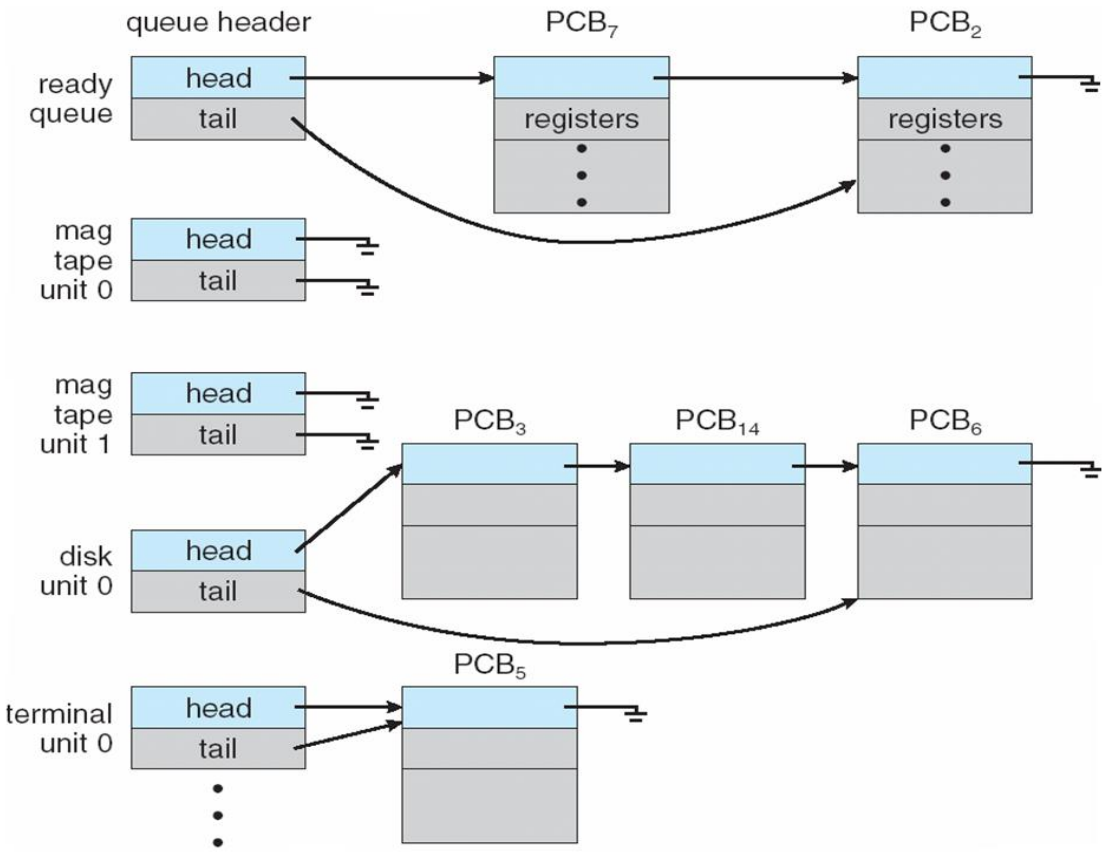


CPU Planlayıcı (Scheduler)

- CPU **bekleme durumuna geçtiğinde** (**Idle**), işletim sistemi hazır kuyruğundan (ready queue) bir prosesi çalıştırılmak üzere **seçmek zorundadır**.
- Seçme işlemi, **kısa dönem planlayıcı (short-term scheduler (CPU scheduler))** tarafından gerçekleştirilir.
- Hazır kuyruğu, **ilk gelen ilk çıkar** (first-in-first-out, FIFO) **olmak zorunda değildir**.
- Hazır kuyruğu, Priority Queue şeklinde de gerçekleştirilmiş olabilir.

CPU Planlayıcı (Scheduler) (devam…)

- Hazır kuyruğunda bekleyen **tüm proseslerin** CPU tarafından çalıştırılma durumları vardır.
- Kuyruk içindeki kayıtlarda, Process Control Block (PCB) tutulur.



Kesintili (preemptive) ve Kesmeyen (non-preemptive) Kavramı

- Hazır kuyruğu ile CPU arasında planlama ilişkisini kuran CPU planlama algoritmaları (CPU scheduling algorithms) temel olarak 2 grupta incelenebilir:
  - **Kesintili algoritmalar (preemptive):** Yürütülen yani **çalışan bir prosesin** CPU'dan kaldırılması ve **istenilen başka bir prosesin (öncelikli)** CPU'da yürütülmesi sağlanabilir.
  - **Kesmeyen algoritmalar (nonpreemptive):** Proses CPU'da çalışmaya başladıktan sonra;
    - Proses **tamamlanıncaya kadar veya ona ayrılan süre kadar** CPU'yu kullanır.
    - Kendi kodunda bulunan bir I/O isteği ile bloklanıncaya kadar çalışır.

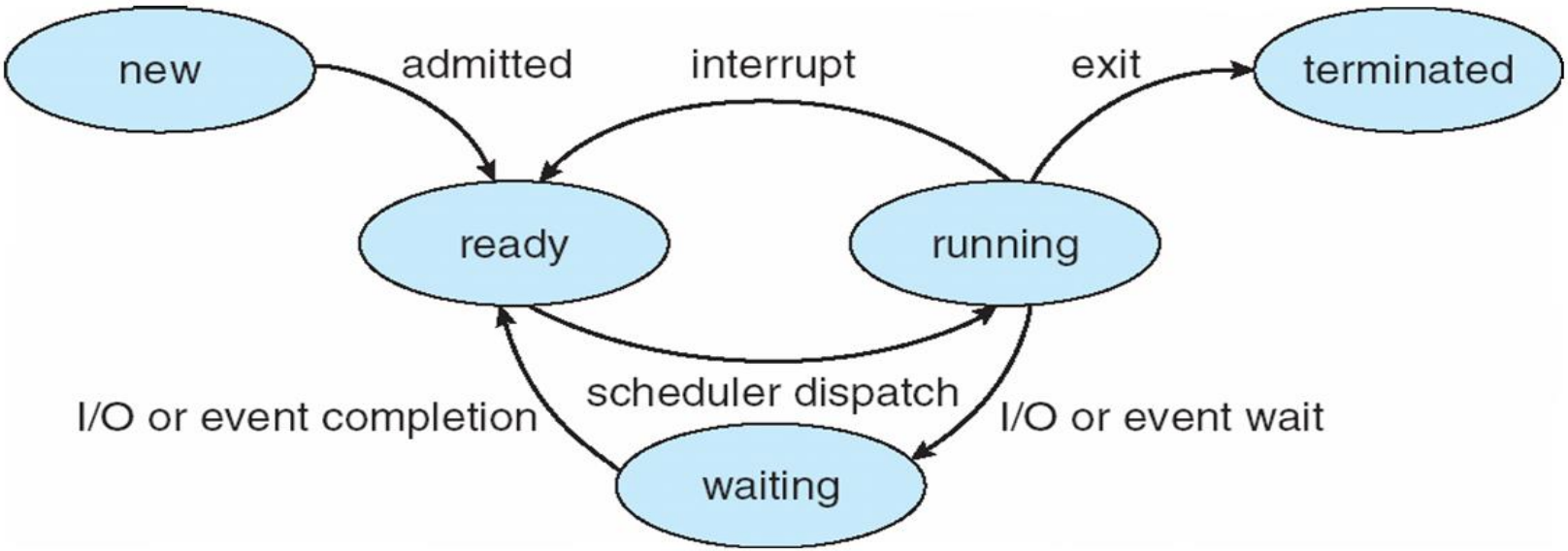
Kesintili (preemptive) ve Kesmeyen (non-preemptive) Kavramı (devam…)

- Windows 3.1, nonpreemptive planlama kullanmıştır.
- **Diğer tüm Windows** versiyonları preemptive planlama kullanmıştır.
- Mac OS X işletim sistemi de preemptive planlama kullanmaktadır.
- Bir prosesin paylaşılan bir veri üzerinde değişiklik yaparken ***yarıda kesilmesi problem oluşturur mu?***

CPU Planlama

- CPU planlama kararı **4 durum altında** gerçekleştirilir:
  1. Bir proses çalışma durumundan bekleme durumuna geçtiğinde (***I/O isteği***),
  2. Bir proses çalışma durumundan hazır durumuna geçtiğinde (***interrupt***),
  3. Bir proses bekleme durumundan hazır durumuna geçtiğinde (***I/O tamamlanması***),
  4. Bir proses **sonlandırıldığında**.
- 1. ve 4. durumlarda planlama açısından **başka seçenek yoktur** ve **yeni bir proses** ***hazır kuyruğundan seçilerek*** çalıştırılmaya başlanır.
- 2. ve 3. durumlarda, şu anki proses direk tekrar CPU'yu alabilir ya da önceliği farklı bir proses CPU'ya yerleştirilir.

Proses Durumları - Hatırlatma



Dağıtıcı/Görevlendirici (Dispatcher)

- İşletim sistemi tasarımında kullanılan görevlendirici, **CPU planlama algoritmasına göre** beklemekte olan proseslerden sıradakini alıp, ***CPU'ya yollayan*** programın ismidir.
- Görevlendirici bu proseslerden **sırası gelenin** hazır kuyruğundan (ready queue) alınarak CPU'ya gönderilmesi işlemini yerine getirir.
- Görevlendiricinin çok hızlı bir şekilde **geçiş yapması zorunludur**.

Planlama (Scheduling) Kriterleri

- CPU planlama algoritmaları çok sayıda farklı kritere göre karşılaştırılır:

1. **CPU kullanım (Utilization):** CPU'nun olabildiği kadar kullanımda olması istenir. CPU kullanım oranı **%0 - %100** arasındadır. Gerçek sistemlerde bu oran **%40 (normal) ile %90 (yoğun)** arasındadır. Utilization = useful time/total time.
2. **Üretilen iş (Throughput):** Birim zamanda tamamlanan proses sayısıdır (saniyede, saatte).
3. **Dönüş süresi (Turnaround time):** Bir prosesin
   - Hafızaya alınmak(HD to RAM) için bekleme süresi,
   - Hazır kuyruğunda bekleme süresi,
   - CPU'da çalıştırılması ve
   - I/O işlemi yapması için **geçen sürelerin toplamıdır**.

Planlama (Scheduling) Kriterleri (devam…)

4. **Bekleme süresi (Waiting time):** Bir prosesin hazır kuyruğunda beklediği süredir.
5. **Cevap süresi (Response time):** Bir prosese istek gönderildikten cevap dönünceye kadar geçen süredir.

Waiting time of a process until it gets the CPU for the first time.

Planlama (Scheduling) Kriterleri (devam…)

Optimizasyon için ne istiyoruz?

- Maksimum CPU Kullanım (utilization)
- Maksimum Üretilen İş (throughput)
- Minimum Dönüş Süresi (turnaround time)
- Minimum Bekleme Süresi (waiting time)
- Minimum Cevap Süresi (response time)

Planlama (Scheduling) Algoritmaları

- CPU planlama algoritmaları, hazır kuyruğunda bekleyen proseslerden hangisinin CPU'ya atanacağını belirlerler.
  1. **First-Come, First-Served (FCFS) Scheduling**
  2. **Shortest-Job-First (SJF) Scheduling**
  3. **Priority Scheduling**
  4. **Round-Robin Scheduling**

1. First-Come, First-Served (FCFS)

- **En basit** CPU planlama algoritmasıdır ve first-come first served (FCFS) şeklinde çalışır.
- CPU'ya **ilk istek yapan proses**, CPU'ya **ilk atanan prosestir**.
- Non-preemptive scheduling algorithm.
- Not priority scheduling algorithm.
- FIFO kuyruk yapısıyla yönetilebilir.
- FCFS algoritmasıyla **ortalama bekleme süresi** genellikle **yüksektir**.
- Bekleme süreleri, proseslerin kuyruğa geliş sırasına göre ***çok değişmektedir***.

1. First-Come, First-Served (FCFS) (devam…)

Proses	Burst Zamanı
P_1	24
P_2	3
P_3	3

- Proseslerin **geliş sırası**: P_1 , P_2 , P_3 bu durumda **Gantt Chart**

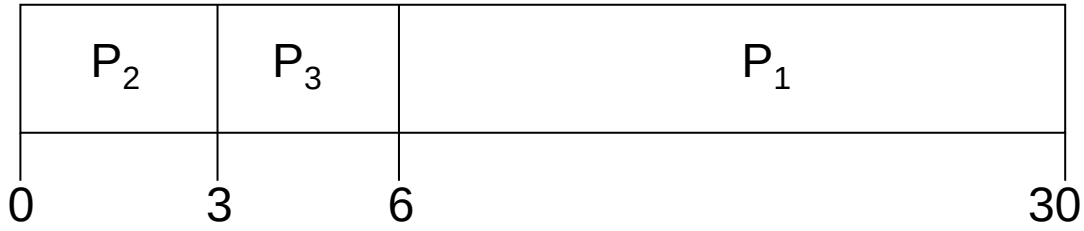
- **Bekleme zamanları:** P_1 = 0; P_2 = 24; P_3 = 27
- **Ortalama bekleme zamanı**: (0 + 24 + 27)/3 = 17 ms

1. First-Come, First-Served (FCFS)
(devam…)

Proseslerin sırası **aşağıdaki gibi değişirse**

$$P_2 , P_3 , P_1$$

- Planlama için Gantt chart



- **Bekleme zamanları:** $P_1 = 6; P_2 = 0, P_3 = 3$
- **Ortalama bekleme zamanı:** $(6 + 0 + 3)/3 = 3$

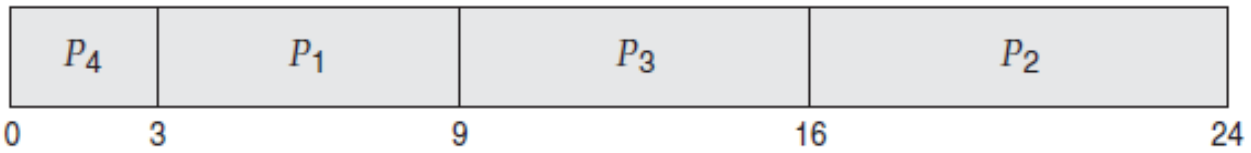
1. First-Come, First-Served (FCFS)
(devam…)

- FCFS algoritmasında, proseslerin çalışma süreleri **çok farklıysa** ortalama bekleme süreleri çok değişken olur.
- **Çok sayıda küçük prosesin** büyük bir prosesin CPU'yu terk etmesini beklemesine **convoy effect** denilmektedir.
- Bir prosese CPU tahsis edildiğinde sonlanana veya I/O isteği gelene kadar CPU'yu elinde tutar.
- FCFS algoritması belirli zaman aralıklarıyla CPU'yu paylaşan **time-sharing sistemler için uygun değildir**.

2. Shortest-Job-First (SJF)

- Shortest-Job-First Scheduling (SJF) algoritmasında, CPU'da bir sonraki **işlem süresi en kısa olan** (shortest-next-CPU-burst olarakta adlandırılır) proses atanır.

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3



- **Ortalama bekleme süresi**, (0 + 3 + 16 + 9) / 4 = 7 ms'dir. FCFS kullanılsaydı 10,25 ms olurdu.
- Burst süresi aynı olan iki adet proses FCFS ile seçilir.

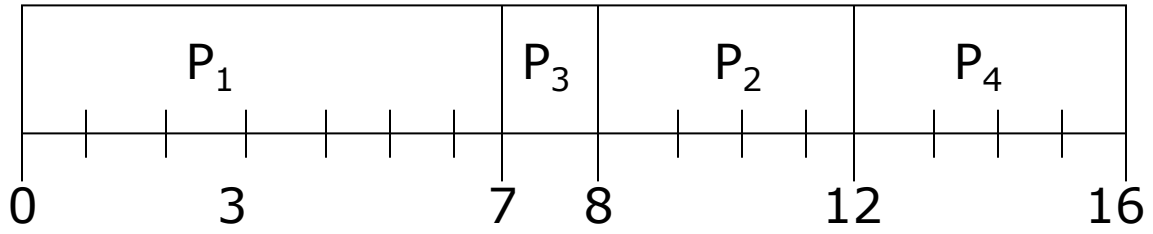
2. Shortest-Job-First (SJF) (devam…)

- SJF algoritması **preemptive** veya **nonpreemptive** olabilir.
- Çalışmakta olan prosesten
  - **Daha kısa süreye sahip yeni bir proses** ready queue'ya geldiğinde,
  - Preemptive SJF **çalışmakta olan prosesi keser**,
  - Non-preemptive SJF çalışmakta olanın sonlanmasına izin verir.
- Preemptive SJF,
  - **Shortest-remaining-time-first** (**Kalan çalışma süresi en kısa olan ilk çalışsın**) planlama olarak adlandırılır.

2. Shortest-Job-First (SJF) (devam…)

Proses	Geliş Zamanı	Burst Zamanı
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (**non-preemptive**)

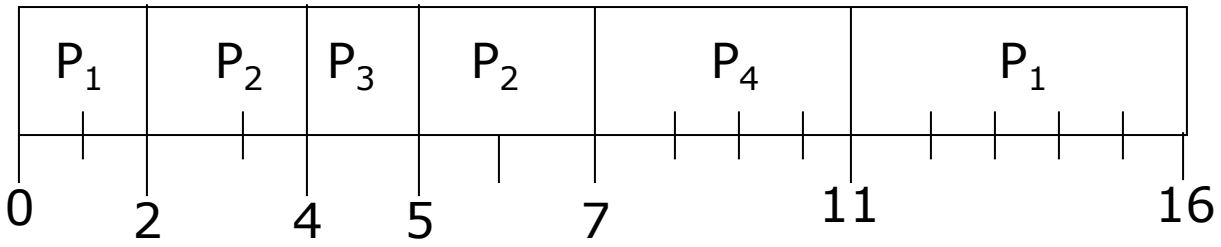


- Ortalama bekleme süresi = (0 + (8 - 2) + (7 - 4) + (12 - 5))/4
- Ortalama bekleme süresi = (0 + 6 + 3 + 7)/4 = 4

2. Shortest-Job-First (SJF) (devam…)

Proses	Geliş Zamanı	Burst Zamanı
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (**preemptive**)

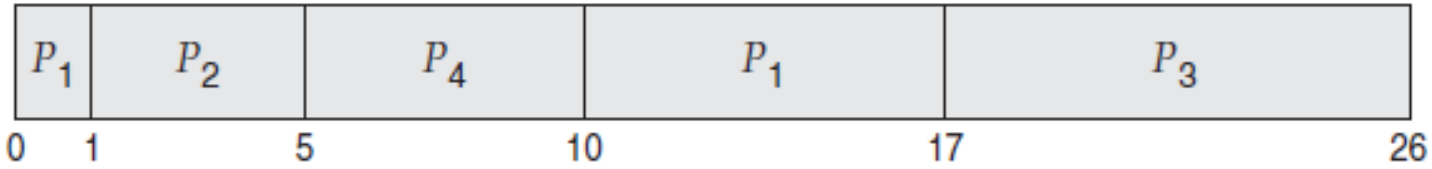


- Ortalama bekleme süresi = (9 + 1 + 0 +2)/4 = 3

2. Shortest-Job-First (SJF) (devam…)

Process	Arrival Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

Preemptive SJF



Ortalama bekleme süresi = [9+0+15+2]/4 = 26/4 = 6.5 msec
Non-preemptive SJF Ortalama bekleme süresi 7.75 msec.

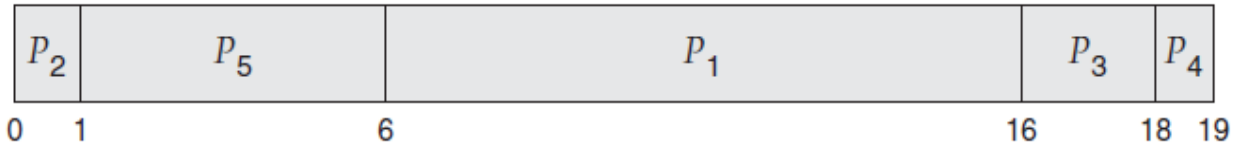
3. Önceliğe Göre Planlama (Priority Scheduling)

- CPU en yüksek önceliğe sahip prosese atanır.
- Eşit önceliğe sahip olanlar ise **FCFS sırasıyla** atanır.
- **Shortest-job-first (SJF)** algoritması, priority planlama algoritmalarının **özel bir durumudur**.
  - SJF algoritması tahmin edilen **CPU-burst süresine göre** önceliklendirme yapar.
  - SJF algoritmasında, **CPU burst süresi azaldıkça öncelik artar**, CPU burst süresi arttıkça **öncelik azalır**.

3. Önceliğe Göre Planlama (Priority Scheduling)

- Aşağıda 5 proses için öncelik değerine göre gantt şeması verilmiştir. Bütün proseslerin 0 zamanında geldiğini düşünelim.

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2



- Ortalama bekleme süresi **(1+ 6 + 16 + 18) / 4 = 8,2 ms** olur.

3. Önceliğe Göre Planlama (Priority Scheduling)

- Önceliklendirme kriterleri aşağıdakilerden bir veya birkaç tanesi olabilir:
  - Zaman sınırı
  - Hafıza gereksinimi
  - Açılan dosya sayısı
  - I/O burst ve CPU burst oranı
  - Prosesin önemi (politik, ödemeler(funds) …)
- Priority planlama **preemptive** veya **nonpreemptive** olabilir.
  - Preemptive yönteminde, bir proses hazır kuyruğuna geldiğinde, çalışmakta olan prosesten daha öncelikli ise, çalışmakta olan kesilir.
  - Nonpreemptive yönteminde, bir process hazır kuyruğuna geldiğinde, çalışmakta olan prosesten daha öncelikli bile olsa, çalışmakta olan devam eder.

3. Önceliğe Göre Planlama (Priority Scheduling)

- Priority planlama algoritmasında,
  - CPU sürekli yüksek öncelikli prosesleri çalıştırabilir ve
  - Bazı prosesler sürekli hazır kuyruğunda bekleyebilir (**indefinite blocking**, **starvation**).
- Sınırsız beklemeyi **engellemek için**
  - Öncelik yaşlanması (**priority aging**) yöntemi kullanılır.
  - Düşük öncelikli prosesler kuyrukta beklerken **öncelik seviyesi artırılır** (Örn. her 15 saniyede 1 artırılır).
- **Öncelik değeri artırılarak** ***en düşük önceliğe sahip*** prosesin bile belirli bir süre **sonunda çalışması** sağlanır.

4. Round Robin

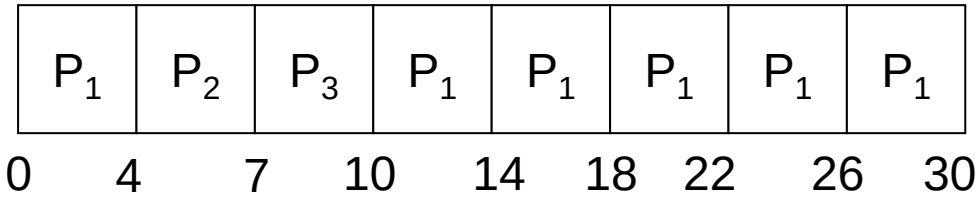
- Round-robin (RR) planlama, genellikle **time-sharing** (zaman paylaşımlı) sistemlerde kullanılır.
- FCFS algoritmasının Preemption eklenmiş halidir.
- Ready queue(circular queue) FIFO olarak işlem görür.
- Hazır kuyruğundaki prosesler **belirli bir zaman aralığında** (*time slice = quantum*) CPU'ya sıralı atanır.
- Zaman aralığı genellikle **«10 ms ile 100 ms»** aralığında seçilir.
  - (1 zaman aralığı = **time quantum**)
- Bu süreden ***daha kısa sürede sonlanan proses*** CPU'yu serbest bırakır.
- Round-robin planlama ile **ortalama bekleme süresi** genellikle uzundur.

4. Round Robin (devam…)

Proses	Süre (Burst Time)
P_1	24
P_2	3
P_3	3

Örnek1 RR:
Time Quantum = 4
Arrival Time = 0

- Gantt Şeması



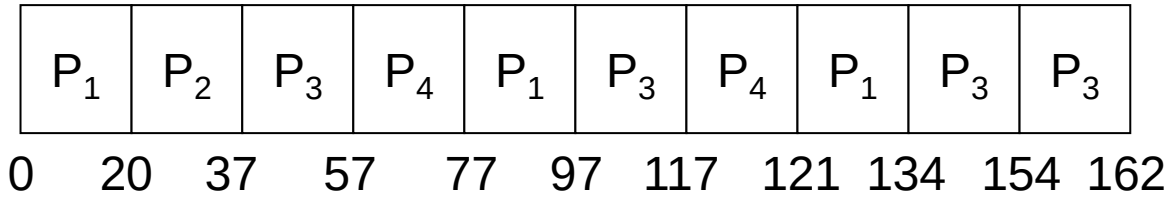
- P1 için bekleme (10-4)=6 , P2=4 , P3=7 bekleme süresine sahiptir.
- ***Ortalama bekleme zamanı***= 17/3=5.66

4. Round Robin (devam…)

Proses	Süre (Burst Time)
P_1	53
P_2	17
P_3	68
P_4	24

Örnek2 RR:
Time Quantum = 20

- Gantt chart:

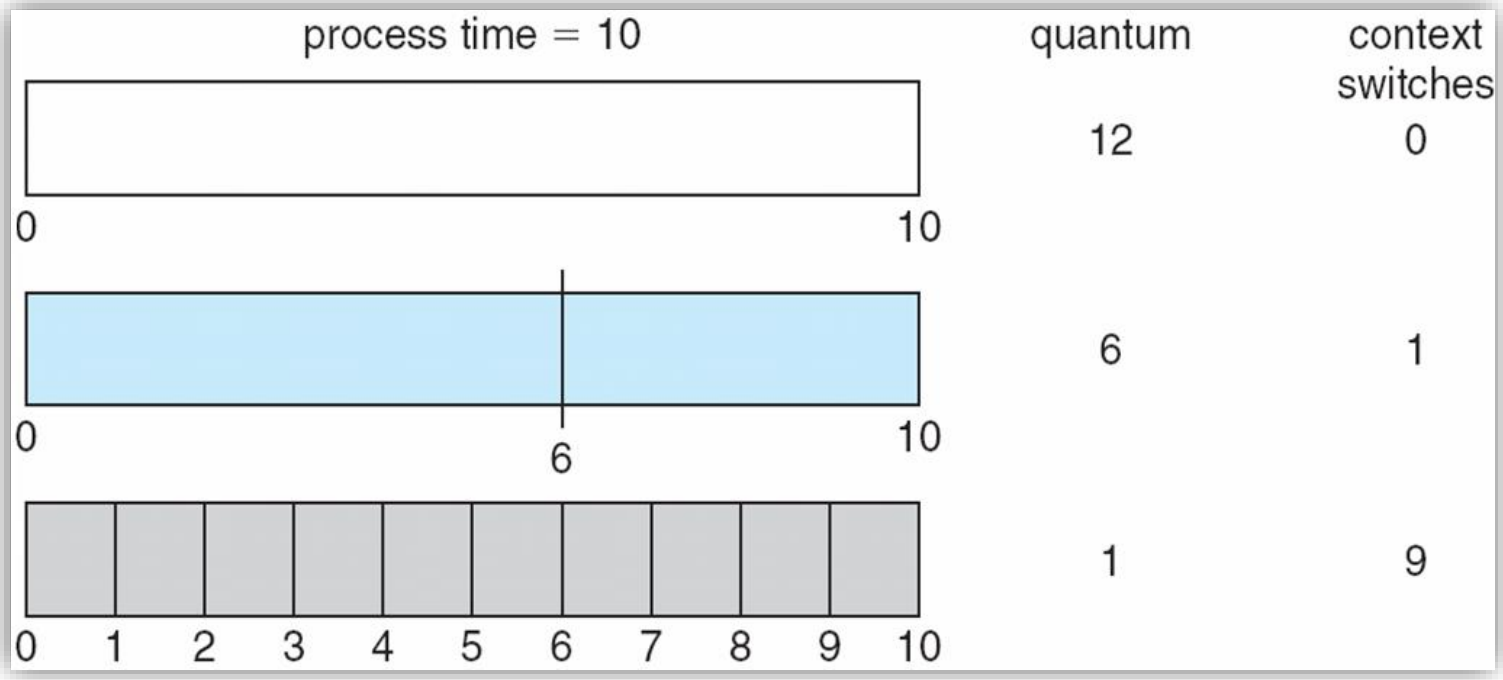


- ***Bekleme zamanı***: P1:57+24=81; P2:20; P3:37+40+17=94; P4:57+40=97

4. Round Robin (devam…)

- Time slice süresi (*quantum*) **çok büyük olursa** çalışma **FCFS** yöntemine benzer.
- Time slice süresi **çok küçük olursa** **context switch** işlemi **çok fazla yapılır**.
  - Context switch süresi **overhead olur** ve çok fazla context switch yapılması ***istenmez***.
- Time slice süresi, context switch süresinin genellikle 10 katı alınır.
  - Modern sistemler quantum 10-100 ms arasında
  - Context switch time <10 ms
  - CPU'nun %10 süresi context switch için harcanır.

4. Round Robin (devam…)



Yararlanılan Kaynaklar

- **Ders Kitabı:**
  - Operating System Concepts, Ninth Edition, Abraham Silberschatz, Peter Bear Galvin, Greg Gagne
- **Yardımcı Okumalar:**
  - İşletim Sistemleri, Ali Saatçi
  - Şirin Karadeniz, Ders Notları
  - İbrahim Türkoğlu, Ders Notları
- **M. Ali Akcayol, Gazi Üniversitesi Bilgisayar Mühendisliği Bölümü**